Government Arts College, Coimbatore(Autonomous)

PG and Research Department of Information Technology

I M.Sc(IT)SOFTWARE TESTING

Sub Code:18MIT32C

Prepared by Dr.P.Radha

Syllabus

UNIT-I: Software Development Life Cycle models: Phases of Software project – Quality, Quality Assurance and Quality control – Testing, Verification and Validation – Process Model to represent Different Phases - Life Cycle models. White-Box Testing: Static Testing – Structural Testing – Challenges in White-Box Testing.

UNIT-II: Black-Box Testing: What is Black-Box Testing - Why Black-Box Testing– When to do BlackBox Testing – How to do Black-Box Testing-Integration Testing: Integration Testing as a Type of Testing – Integration Testing as a Phase of Testing – Scenario Testing – Defect Bash.

UNIT-III: System and Acceptance Testing: System Testing Overview – Why is System Testing done – Functional versus Non-functional Testing – Functional system Testing - Non-functional Testing – Acceptance Testing – Summary of Testing Phases.

UNIT-IV: Performance Testing: Factors Governing Performance Testing – Methodology for Performance Testing – Tools for Performance Testing – Process for Performance Testing – Challenges. Regression Testing: What is Regression Testing – Types of Regression Testing – When to do Regression Testing – How to do Regression Testing– Best Practices in Regression Testing.

UNIT-V:Test Planning, Management, Execution and Reporting: Test Planning – Test Management – Test Process – Test Reporting –Best Practices. Test Metrics and Measurements: Project Metrics – Progress Metrics – Productivity Metrics – Release Metrics.

UNIT I

Software Development Life Cycle models: Phases of Software project – Quality, Quality Assurance and Quality control – Testing, Verification and Validation – Process Model to represent Different Phases - Life Cycle models. White-Box Testing: Static Testing – Structural Testing – Challenges in White-Box Testing.

TEXT BOOK: “SOFTWARE TESTING Principles and Practices”-Srinivasan Desikan & Gopalswamy Ramesh, 2006, Pearson Education

Prepared by Dr.P.Radha

Software Development Life Cycle models

- Phases of Software project
- Quality, Quality Assurance and Quality control
- Testing, Verification and Validation
- Process Model to represent Different Phases
- Life Cycle models.

Phases of Software project

A software project is made up of a series of phases:

- Requirements Gathering and Analysis
- Planning
- Design
- Development or Coding
- Testing
- Deployment or Maintenance

Quality, Quality Assurance and Quality control

Quality is meeting the requirements expected of the software, consistently and predictably.

Each test case is further characterized by

- The environment under which the test case is to be executed;
- Inputs that should be provided for that test case;
- What changes should be produced in the internal state or environment ;and
- What output should be produced.

- Quality Control is defect-detection and defect-correction oriented, and works on the product rather than on the process.
- Quality Assurance, attempts defect prevention by concentrating on the process of producing rather than working on defect detection/correction after the product is built.

Testing, Verification and Validation

- The purpose of testing is to uncover defects in the system (and to have someone fix the defects).
- Testing is done by a set of people within a software product (or service) organization whose goal and charter is to uncover the defects in the product before it reaches the customer.

Verification

Verification is the process of evaluating a system or component to determine whether the products of a given phase satisfy the conditions imposed at the start of that phase.

Validation

Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.

Quality Assurance=Verification

Quality Control = Validation

= Testing

Process Model to represent different Phases

In this model, each phase of a software project is characterized by the following.

- Entry criteria, which specify when that phase can be started. Also included are the inputs for the phase.
- Tasks, or steps that need to be carried out in that phase, along with measurements that characterize the tasks.
- Verification ,which specifies methods of checking that the tasks have been carried out correctly.
- Exit criteria ,which stipulate the conditions under which one can consider the phase as done. Also included are the outputs for only the phase.

Cont…

- This model, known as the Entry Task Verification eXit or ETVX model, offers several advantages for effective verification and validation.

ETVX model applied to design

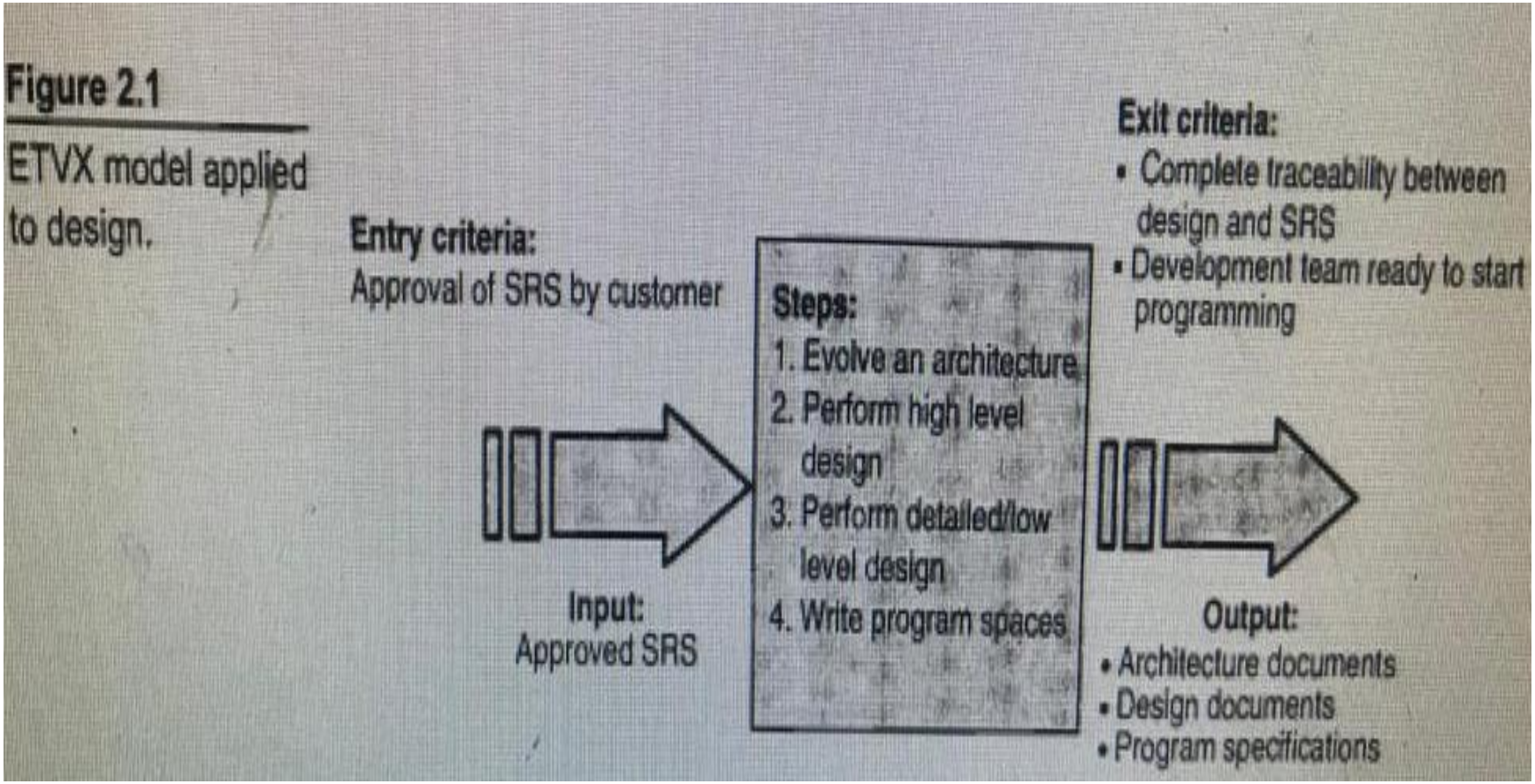


Figure 2.1

ETVX model applied to design.

Life Cycle Models

The ETVX model characterizes a phase of a project. A Life Cycle model describes how the phases combine together to form a complete project or life cycle. Such a model is characterized by the following attributes.

- The activities performed
- The deliverables from each activity
- Methods of validation of the deliverables
- The sequence of activities
- Methods of verification of each activity, including the mechanism of communication amongst the activities

Life Cycle Models

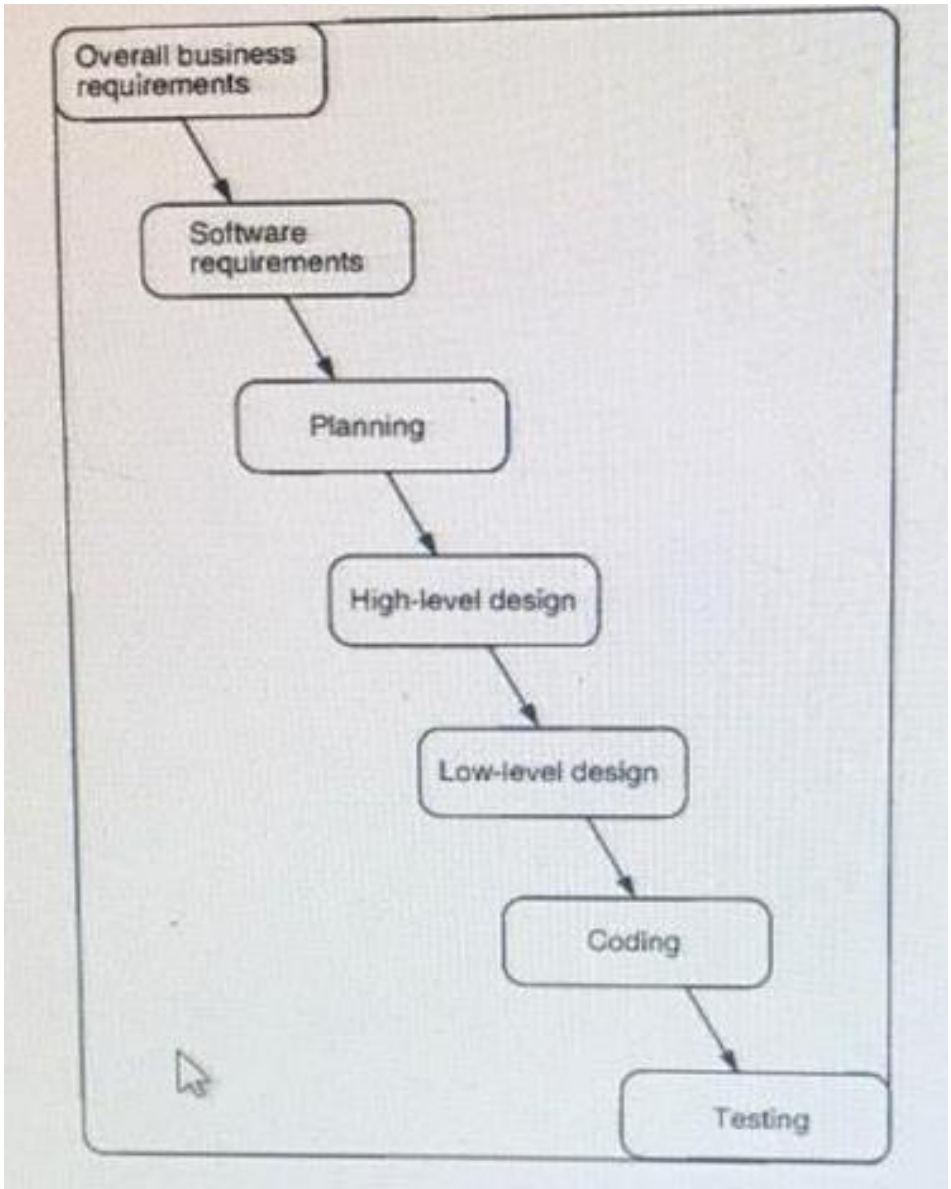
- Waterfall Model
- Prototyping and Rapid Application Development Models
- Spiral or Iterative Model
- The V Model
- Modified V Model

Waterfall Model

In the Waterfall model, a project is divided into a set of phases (or activities).

- Overall business requirements
- Software requirements
- Planning
- High-level design
- Low-level design
- Coding
- Testing

Water Fall Model



Cont..

Waterfall Model is characterized by three attributes

1.The project is divided into separate distinct phases.

2.Each phase communicates to the next through pre-specified outputs.

3.When an error is detected, it is traced back to one previous phase at a time, until it gets resolved at some earlier phase.

Prototyping and Rapid Application Development Models

Prototyping and Rapid Application Development (RAD)models recognize and address the following issues.

1. Early and frequent user feedback will increase the chances of a software project meeting the customers' requirements.

2. Changes are unavoidable and the software development process must be able to adapt itself to rapid changes.

Cont…

1. Prototyping model uses constant user interaction, early in the requirements gathering stage to produce a prototype.
2. The prototype is used to derive the system requirements specification and can be discarded after the SRS is built.
3. An appropriate life cycle model is chosen for building the actual product after the user accepts the SRS.

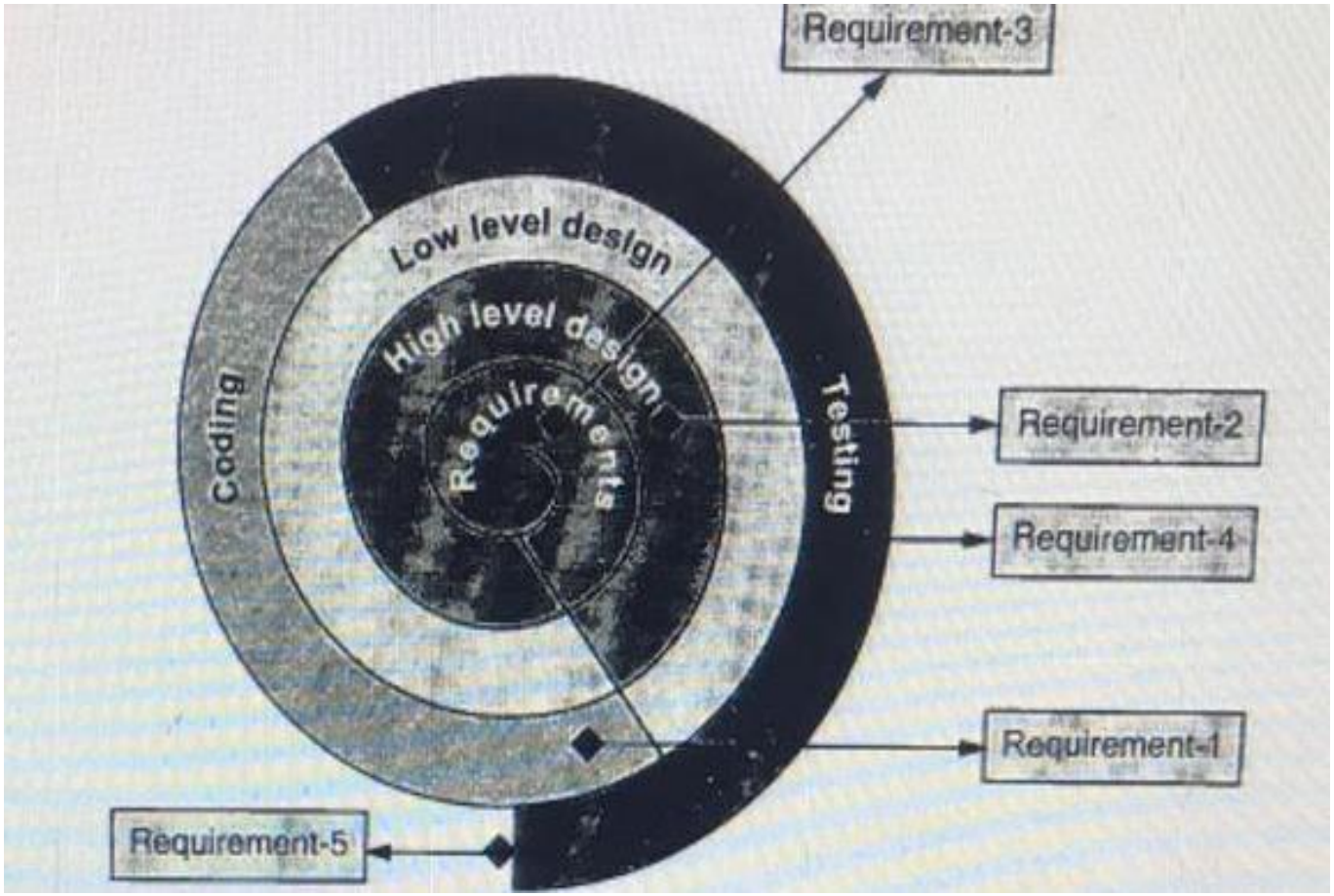
RAD MODEL

- It is not a prototype that is built but the actual product itself.
- In order to ensure formalism in capturing the requirements and proper reflection of the requirements in the design and subsequent phases, a CASE tool is used throughout the lifecycle, right from the requirements gathering.

Spiral or Iterative Model

The progress of the product can be seen from the beginning of the project as the model delivers "increments" at regular intervals. Even though it will be very difficult to plan a release date following this model, it allows the progress to be tracked and the customer approvals to be obtained at regular intervals, thereby reducing the risk of finding major defects at a later point of time.

Cont..

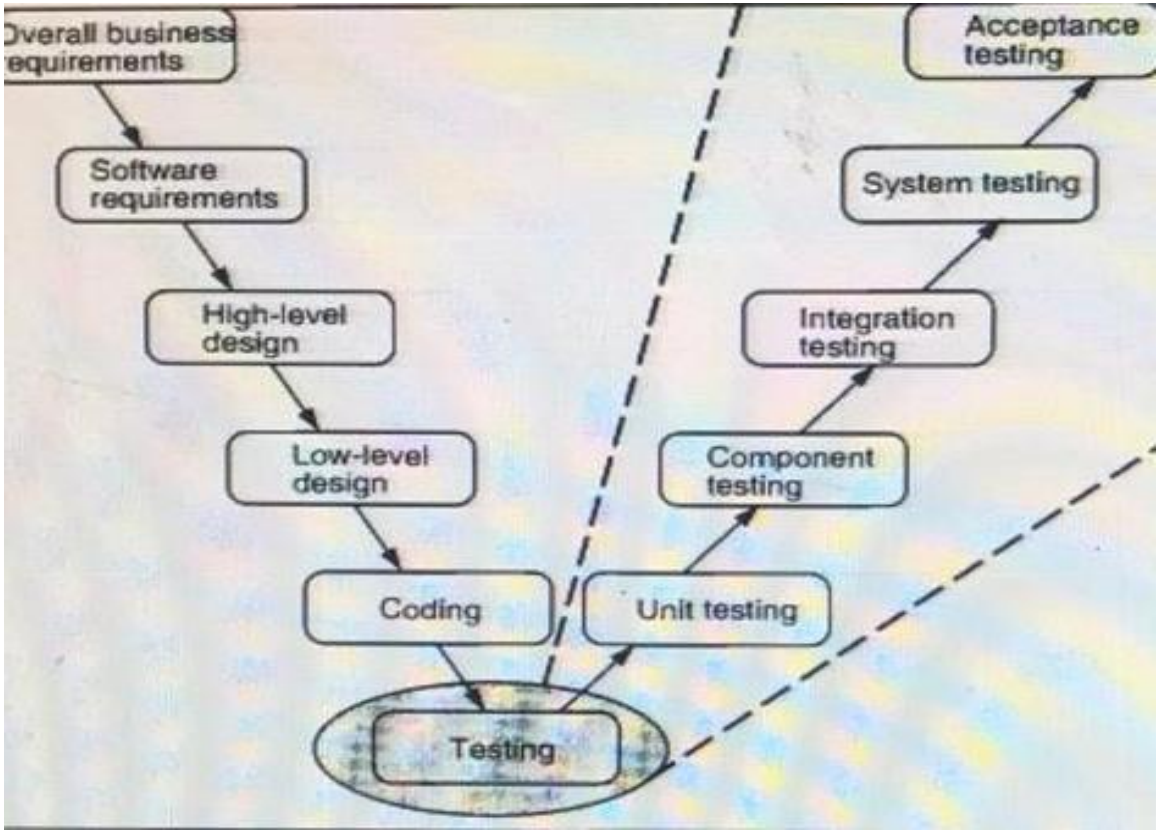

Requirement-3
Low level design
High level design
Requirements
Coding
Testing
Requirement-2
Requirement-4
Requirement-1
Requirement-5

The V Model

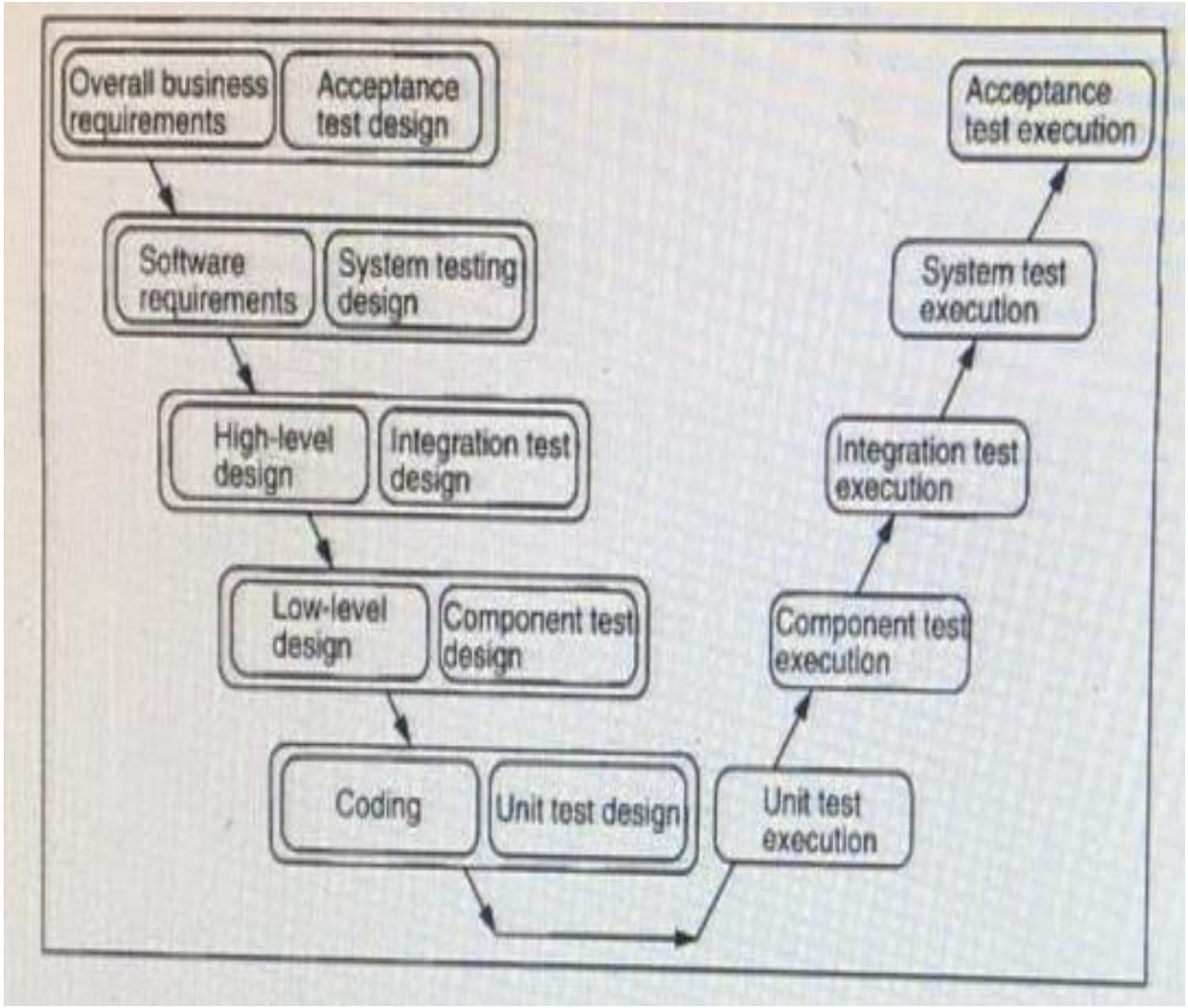
The V Model starts off being similar to the Waterfall Model in that it envisages product development to be made up of a number of phases or levels.

1. The V-model splits testing into two parts-design and execution.
2. Test design is done early, while test execution is done in the end.
3. There are different types of tests for each phase of life cycle.

Phases of testing for different development phases



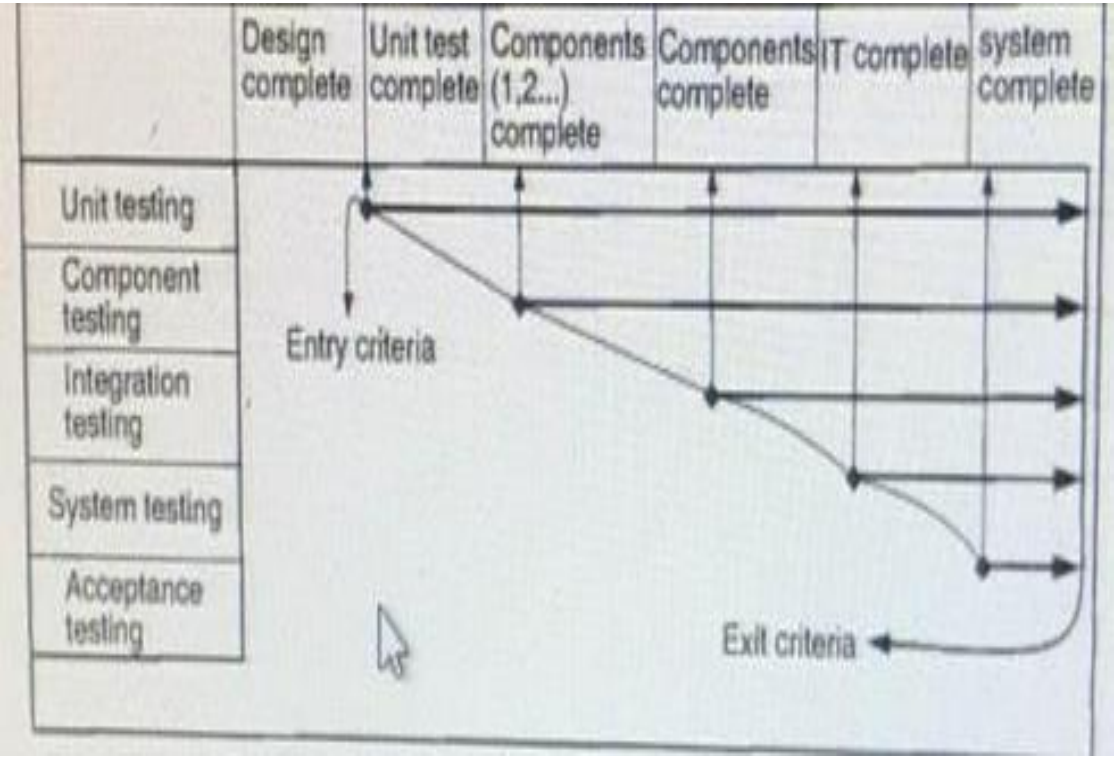
V-Model


Overall business requirements
Acceptance test design
Software requirements
System testing design
High-level design
Integration test design
Low-level design
Component test design
Coding
Unit test design
Unit test execution
Component test execution
Integration test execution
System test execution
Acceptance test execution

Modified V Model

1. The modified V model recognizes that different parts of a product are in different stages of evolution.
2. Each part enters the appropriate testing phase(such as unit testing, component testing, and so on)when the appropriate entry criteria are met.

Modified V Model



White-Box Testing

- Static Testing
- Structural Testing
- Challenges in White-Box Testing

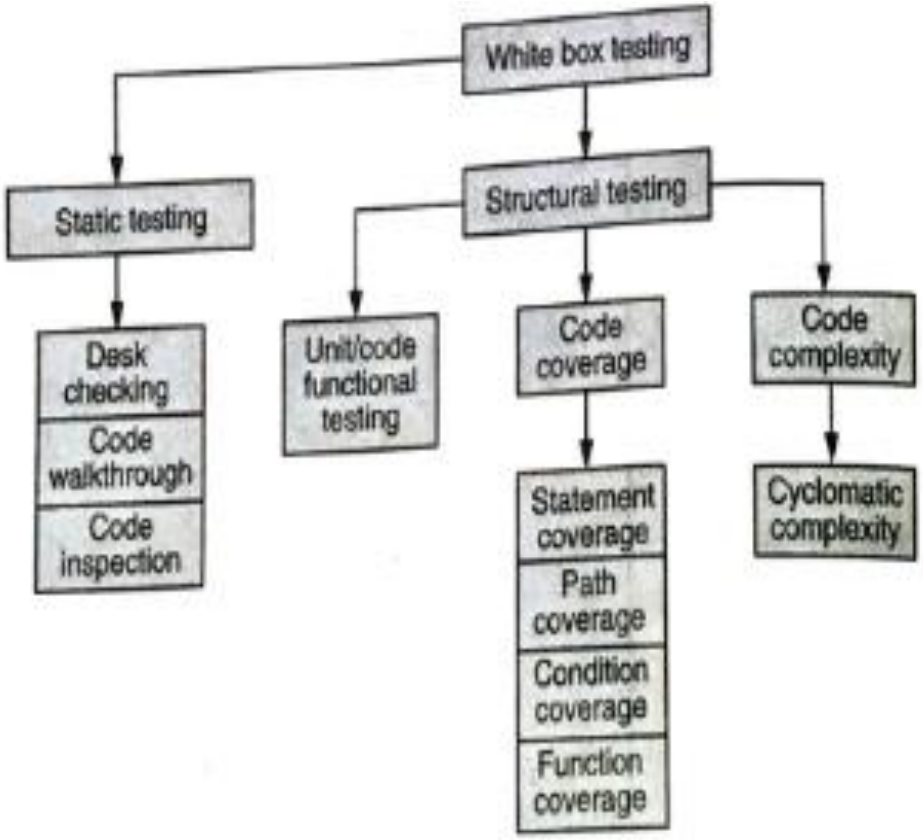
WHAT IS WHITE BOX TESTING?

- White box testing is a way of testing the external functionality of the code by examining and testing the program code that realizes the external functionality.
- This is also known as clear box, or glass box or open box testing.

Cont…

- White box testing takes into account the program code, code structure, and internal design flow

Classification of white box testing



Static Testing

Static testing is a type of testing which requires only the source code of the product, not the binaries or executables.

Cont…

Static testing does not involve Static testing executing the programs on computers but involves select people going through the code to find out whether

- The code works according to the functional requirement;
- The code has been written in accordance with the design developed earlier in the project lifecycle;
- The code for any functionality has been missed out;
- The code handles errors properly.

Static Testing by Humans

There are multiple methods to achieve static testing by humans. They are

- Desk checking
- Code walkthrough
- Code review
- Code inspection

Static Analysis Tools

There are several static analysis tools available in the market that can reduce the manual work and perform analysis of the code to find out errors such as those listed below:

- Whether there are unreachable codes (usage of GOTO statements some times creates this situation; there could be other reasons too)

Cont…

- Variables declared but not used
- Mismatch in definition and assignment of values to variables
- Illegal or error prone type casting of variables

  Use of non-portable or architecture-dependent programming constructs
- Memory allocated but not having corresponding Statements for freeing them up memory
- Calculation of cyclomatic complexity

STRUCTURAL TESTING

- Structural testing takes into account the code, code structure, internal design, and how they are coded.
- The fundamental difference between structural testing and static testing is that in structural testing tests are actually run by the computer on the built product, where as in static testing, the product is tested by humans using just the source code and not the executables or binaries.

STRUCTURAL TESTING

- Unit/Code Functional Testing
- Code Coverage Testing
  - Statement Coverage
  - Path Coverage
  - Condition Coverage
  - Function Coverage
- Code Complexity Testing

Challenges in White box testing

- White box testing requires a sound knowledge of the program code and the programming language.
- Human tendency of a developer being unable to find the defects in his or her code.
- Fully tested code may not correspond to realistic scenarios.